

Error Handling 1 - Axis Errors

The motion coordinator controllers are designed to trap error conditions in hardware, and if required to automatically open the drive enable relay (watchdog) and to disable the output to the drives.

As this mechanism happens automatically, it may not be immediately apparent that an error has occurred and therefore we need a mechanism in the software to recognise it, and to set up the type of errors which will cause the controller to disable the drive / output.

The relevant parameters are:

- AXISSTATUS
- ERRORMASK
- MOTION_ERROR
- ERROR_AXIS

AXISSTATUS

The AXISSTATUS axis parameter may be used to check various status bits held for each axis fitted:

Example:

```
IF (AXISSTATUS AND 16)>0 THEN PRINT "In forward limit"
```

Bit	Description:	Value:
0	Unused	1
1	Following error warning range	2
2	Unused	4
3	Unused	8
4	In forward limit	16
5	In reverse limit	32
6	Datuming	64
7	Feedhold	128
8	Following error exceeds limit	256
9	In forward software limit	512
10	In reverse software limit	1024
11	Cancelling move	2048

ERRORMASK

The value held in this parameter is bitwise ANDed with the AXISSTATUS parameter by every axis on every servo cycle to determine if a runtime error should switch off the enable (WDOG) relay. If the result of the AND operation is not zero the enable relay is switched OFF. The default setting is 256. This will trip the enable relay only if a following error condition occurs.

MOTION_ERROR

This system parameter has the value 1 when some axis has had a motion error, i.e. following error, and the value 0 when no axis has had a motion error. When there is a motion error then the ERROR_AXIS contains the number of the first axis to have an error. When any axis has a motion error then the WDOG relay is opened. A motion error can be cleared by resetting the controller with an EX command ("Reset the controller." under the "Controller" menu in Motion perfect), or by using the DATUM(0) command.

ERROR_AXIS

Returns the number of the axis which caused the enable WDOG relay to open when a following error exceeded its limit.

Description

This example task deals only with the safety logic issues and assumes that another task (i.e. STARTUP) is used to setup and reset the axes after an error has occurred.

```

O . start: . O
. .   estop_state=0 .
O .   WAIT UNTIL IN(7)=ON ' ESTOP reset enabled . O
. .   ' Monitor constantly until ESTOP state changes or .
O .   ' axis error occurs . O
. . . . .
O . REPEAT . O
. .   estop_state=IN(7) .
O . UNTIL( estop_state=0 OR MOTION_ERROR<>0) . O
. .   ' If here then MUST be either Estopped, .
O .   ' axis error or both . O
. . . . .
O .   ' Stop other tasks, e.g. . O
. .   '   STOP "main" .
O .   '   STOP "motion" . O
. . . . .
O . WDOG=OFF . O
. . . . .
O . IF MOTION_ERROR<>0 THEN . O
O . ax = ERROR_AXIS . O
. . BASE(ax) .
O . PRINT #3,CURSOR(0);"Error on Axis ";ax[0] . O
. .   IF (AXISSTATUS AND 256)>0 THEN PRINT #3,"Following Error"; .
O .   IF (AXISSTATUS AND (16+32))>0 THEN PRINT #3,"Hard Limit"; . O
O .   IF (AXISSTATUS AND 512+1024)>0 THEN PRINT #3,"Soft Limit"; . O
. . ENDIF .
O . . . . .
O . IF estop_state=0 THEN . O
. .   PRINT #3,"EMERGENCY STOP" .
O .   PRINT #3,"Press any key:" . O
. . ENDIF .
O . . . . .
O . WAIT UNTIL KEY#3 . O
O . GET #3,k . O
. . . . .
O . GOTO start . O
. . . . .

```